PARAMETER PASSING OF DATA STRUCTURES WHERE API AND CORRESPONDING STORED PROCEDURE ARE DIFFERENT VERSIONS/RELEASES

Background of the Invention

1. Cross Reference to Related Application

[0001] This application is related to co-pending U.S. Patent Application Serial No. 10/128,260 titled "Method and Apparatus of Parameter Passing of Structured Data for Stored Procedures in a Content Management System," which is assigned to the same assignee as the present application.

2. Field of the Invention

[0002] The present invention is directed to improvements in computing systems and in particular to improvements in operability between application programming interfaces (APIs) and stored procedures interacting with the APIs when software upgrades or modifications are made to either one or both of the API and/or stored procedures.

3. Description of the Prior Art

[0003] Stored procedures are collections of pre-defined procedural code that are used to perform one or more tasks such as, for example, to access databases. There are a number of benefits in using stored procedures, including function encapsulation, performance enhancement,

client-server processing, and security. Generally, a user explicitly invokes stored procedures in an interactive environment, but such procedures can also be invoked by other programs.

[0004] In today's software systems, an application may rely one or more of the stored procedures for various services. An application interfaces with the stored procedures or other applications through an application program interface (API). Such application integration allows an application to provide services without having to include the programming and data necessary to provide the services itself.

[0005] For example, a content management system (CMS) may rely on a database application for maintaining large amounts of information such as binary large objects (BLOBs) and character large objects (CLOBs). Stored procedures are utilized by the CMS to efficiently communicate with a database management system (DBMS). The CMS communicates with the stored procedures via a set of APIs in order to access the database information. The APIs are typically built in their own dynamic link library (DLL). The CMS may load the provided DLL and use the appropriate APIs for maintaining the BLOBs, CLOBs and other information stored in the database.

[0006] Software is typically upgraded periodically to include new or improved features. In the CMS example, the CMS and stored procedures may be independently updated in scheduled releases of new versions or maintenance levels. Each new update or release may include changes to the respective APIs, and the updates may be sent directly to a plurality of users. One problem that can result is that the updated APIs may not be inter-compatible between the CMS and the stored procedures when the CMS and stored procedures are at a different version or level with respect to each other. Such incompatibility may cause the CMS or the

stored procedures to function incorrectly or not to function at all. For example, a stored procedure may only handle primitive data types such as integer, long, BLOB, and CLOB in the parameter passing portion of its interface. The stored procedure typically does not handle structured data of variable length such as, for example, an array of integers, an array of character strings or an array of mixed data types in the parameter passing of its interface.

[0007] To illustrate the problem further, in Content Manager Version 8 Release 1, an IBM product, this problem was solved with a pair of parameters (BLOB and CLOB), where the BLOB contained numeric values representing the:

- data element identifier,
- data element type (i.e., small integer, integer, character),
- data element length, and
- data element value (for numeric data elements only).

[0008] The CLOB contained the data element values of character elements. Common routines are invoked by both of the APIs and respective stored procedures to build and parse the BLOB/CLOB pair.

[0009] However, with a second release of Content Manager Version 8 (V8.2), it was necessary to add data elements to data structures being passed in BLOB/CLOB pairs. A problem is encountered because many customers can be running environments where APIs are at a different version/release levels than their stored procedures (e.g., the APIs are at level V8.1 and the stored procedures are at V8.2 or higher, and vice versa). There are two scenarios which present two distinct problems. In the first scenario, the APIs are at a lower version/release level

than the stored procedures. In this first scenario, the stored procedures expect additional data elements at the end of a data structure, and attempt to parse them, but they are not there because the V8.1 APIs do not pass them, not having any knowledge of them.

[0010] In the second scenario, the APIs are at a higher version/release level than the stored procedures. In this scenario, the APIs pass additional data elements, elements of which the stored procedures have no knowledge. Both scenarios result in a parsing error according to the current method of building/parsing BLOB/CLOB pairs. This presents a continuing problem for future releases as elements are appended to data structures.

[0011] It is therefore desirable to provide an improved method and means of passing parameters for data structures where the API and the respective stored procedures are at different version or release levels.

Brief Summary of the Invention

[0012] In accordance with the present invention, there is provided a method of parameter passing of data structures where an API and corresponding stored procedures are at different version/release levels. The method includes receiving a data structure comprising data structure elements from a caller and parsing the data structure for a version identifier. The parsed version identifier is compared to a stored procedure version identifier. If the comparison is indicative of a data structure compatibility between the calling program and the stored procedures, all received data structure elements are parsed. If the comparison is indicative of a data structure incompatibility between the calling program and the stored procedures, only data structure elements known to both of the calling program and the stored procedures are parsed.

[0013] In accordance with another aspect of the present invention, there is provided a system for passing parameters of data structures where an API and corresponding stored procedures are at different version/release levels. A means is provided for receiving a data structure comprising data structure elements from a calling program and parsing the data structure for a version identifier. A comparison means compares the parsed version identifier to a stored procedure version identifier. If the comparison means determines a data structure compatibility between the calling program and the stored procedures, a parsing means parses all received data structure elements. If, however, the comparison means determines a data structure incompatibility between the calling program and the stored procedures, the parsing means parses only data structure elements known to both of the calling program and the stored procedures.

[0014] In accordance with still another aspect of the present invention, there is provided a computer program product having means contents for passing parameters of data structures where an API and corresponding stored procedures are at different version/release levels. Program code is provided for receiving a data structure comprising data structure elements from a calling program and parsing the data structure for a version identifier. Other program code compares the parsed version identifier to a stored procedure version identifier. If the comparison program code determines a data structure compatibility between the calling program and the stored procedures, parsing program code parses all received data structure elements. If, however, the comparison program code determines a data structure incompatibility between the calling program and the stored procedures, the parsing program code parses only data structure elements known to both of the calling program and the stored procedures.

[0015] One advantage obtained from the present invention is the elimination of parsing errors when a version/release level of a set of APIs is different than the version/release level of a corresponding set of stored procedures.

[0016] Another advantage obtained from the present invention is the reduction in computer system program maintenance necessary when upgrading with a set of APIs or a set of stored procedures.

[0017] Other advantages of the subject method and system will become apparent to those skilled in the art upon a reading and understanding of this specification.

Brief Description of the Drawings

[0018] The invention may take physical form in certain parts and steps and arrangements of parts and steps, the embodiments of which will be described in detail in this specification and illustrated in the accompanying drawings hereof and wherein:

[0019] FIG. 1 is a block diagram of an exemplary environment suitable for practicing aspects of the present invention;

[0020] FIG. 2 is an illustration of an exemplary BLOB/CLOB pair suitable for practicing aspects of the present invention; and

[0021] FIG. 3 is a flowchart of a method according to the present invention.

Detailed Description of the Invention

[0022] Reference will now be made in detail to an embodiment of the present invention, examples of which are illustrated in the accompanying drawings. The detailed description which follows is presented in terms of general procedures, steps and symbolic

representations of operations of data bits within a computer memory, associated computer processors, networks, and network devices. These procedure descriptions and representations are the means used by those skilled in the data processing art to convey the substance of their work to others skilled in the art. A procedure is here, and generally, conceived to be a self-consistent sequence of steps or actions leading to a desired result. Thus, the term "procedure" is generally used to refer to a series of operations performed by a processor, be it a central processing unit of a computer, or a processing unit of a network device, and as such, encompasses such terms of art as "objects," "functions," "subroutines" and "programs."

[0023] The procedures presented herein are not inherently related to any particular computer or other apparatus. In particular, various general purpose machines may be used with programs in accordance with the teachings herein, or it may prove more convenient to construct more specialized apparatus to perform the required method steps.

[0024] However, one of ordinary skill in the art will recognize that there exists a variety of platforms and languages for creating software for performing the procedures outlined herein. One of ordinary skill in the art also recognizes that the choice of the exact platform and language is often dictated by the specifics of the actual system constructed, such that what may work for one type of general purpose computer may not be efficient on another type of general purpose computer.

[0025] One of ordinary skill in the art to which this invention belongs will have a solid understanding of content management systems, application programming interfaces, stored procedures, and methods of accessing and storing items managed by a content management system. It being recognized that such practitioners do not require specific details of the software,

but rather find data structure descriptions and process descriptions more desirable (due to the variety of suitable hardware and software platforms), such specifics are not discussed to avoid obscuring the invention.

[0026] FIG. 1 is a block diagram that illustrates an exemplary environment suitable for practicing concepts of the present invention. A user application **10**, such as a CMS for example, utilizes APIs **12** for communicating with stored procedures **14**. The stored procedures **14** and the APIs **12** may be on the same computer system or may be on separate systems, in communication via a network connection (not shown). Concepts of the present invention are not limited in scope to a particular type of system, and the present invention may be realized on systems ranging from a single computer to a distributed system operating over a plurality of network connections. The stored procedures **14** are typically in communication with a DBMS **16** for accessing a database **18** where managed content is maintained.

[0027] With reference now to FIG. 2, and continuing reference to FIG. 1, in the embodiment shown, the APIs **12** are used by the CMS **10** to send and receive BLOBs **20** and CLOBs **22** to and from the stored procedures **14** for maintaining information in the database **18** via the DBMS **16**. A version/release identifier **24** of the API **12** is placed at the start of each BLOB/CLOB pair **20/22**. The stored procedures **14** first parse the version/release identifier **24** placed at the start of each BLOB/CLOB pair, and then take the appropriate action to parse only what is expected for that version/release of the API **12**, in other words, it parses only what is known to the API. Likewise, when one of the APIs **12** is parsing a BLOB/CLOB pair **20/22** which is returned from one of the stored procedures **14**, it checks a Library System Control Table **25** to obtain the version/release number of the stored procedure, and then parses according to

rules in effect for the obtained version/release number of the stored procedure, in other words, it parses only what is known to the stored procedure.

[0028] The exemplary BLOB **20** shown in the figure comprises a first data element ID **26**, a first data type **28**, a first data length **30**, a second data element ID **32**, a second data type **34** and a second data length **36**. These describe, respectively, a first character data value **38** and a second character data value **40** that form the exemplary CLOB **22**. The illustrated BLOB/CLOB pair **20/22** is for exemplary purposes only, and the BLOB **20** may define any number of data elements, by means of any suitable coding method, not limited by the example illustrated. The CLOB **22**, likewise, may define any number of data values.

[0029] Provision is made for the case where the parsing component, either a calling API **12** program or a receiving stored procedure **14** program, is an earlier version/release than the other. In exemplary embodiments, a common parse routine **42** automatically skips data elements which the receiving API or stored procedure is not expecting, and searches ahead in the data structure for the next tuple (data element and data value) which matches what the parsing component (API or stored procedure) expects or knows about. This technique avoids the necessity of incorporating this logic in every stored procedure and API. In the exemplary BLOB/CLOB pair **20/22** for example, data element #1 **26-30, 38** was incorporated in an early version of a CMS, and data element #2 **32-36, 40** was incorporated into a later version of the CMS, so that data element #2 would be unknown to implementations of the early version of the CMS and the respective APIs.

[0030] Special consideration is given, as well, to the scenarios where either of the APIs or stored procedures are at an early version level that does not incorporate concepts of the

present invention. The early code may be generally available, and it is thus a goal to not require a change to every API and stored procedure in order to add version/release information to every BLOB/CLOB pair in those instances where they have not changed. The absence of the version/release information at the start of the BLOB/CLOB pair will serve as an indication that the component is at an early version/release level not incorporating concepts of the present invention.

[0031] Although the preferred embodiments disclosed herein describe a numerical comparison of version/release numbers, the invention also includes the case wherein the version/release numbers of the APIs and the version/release numbers of the respective stored procedures are not numbered according to the same schedule or scheme. For example, it may be the case that a version 5.0 API is compatible with version 10.0 stored procedures, or vice versa. In this case, rather than simply comparing version/release numbers numerically, embodiments of the present invention determine compatibility by mapping API version/release numbers to compatible, or incompatible, stored procedure version release numbers. This is accomplished by various methods known in the art, such as, for instance, a system version mapping table. This also proves beneficial in the case where an older version release of an API is compatible with a newer version/release of the stored procedures, and vice versa.

[0032] With reference now to FIG. 3, a method **50** for parameter passing of data structures is described that is sufficient to cover two scenarios, a first where the API is at a lower version/release than the stored procedures, and a second where the API is at a higher version/release than the stored procedures. In describing the method, reference is made to a caller, where the caller is one of the API of the user application or the CMS. In a first step **52**, the

received BLOB/CLOB pair is parsed to ascertain the version and release information, or absence of same, of the caller. A first comparison **54** is performed to determine if the caller's version and release is identical to that of the stored procedure. If the comparison is true, the version and release information matches, then parsing of character data elements known to both the caller and the stored procedure occurs at step **56**. In the example of FIG. 2, character data elements #1 and #2 are parsed.

[0033] In the case where the comparison **54** is not true, the version and release do not match, or there is no version/release information present in the received BLOB, a determination **58** is made as to whether or not the caller's version and release is lower (earlier) than that of the stored procedure. If the caller's version/release is lower, or absent, processing continues at step **60** where only the character data elements known to the caller are parsed. In the example, only character data element #1 is parsed. This avoids errors that would occur if the stored procedure would expect additional data elements not known to the caller which is at an earlier version/release than the stored procedure.

[0034] If the caller's version/release is at a higher value (later) than that of the stored procedure, only character data elements known to the stored procedure are parsed. At step **62** a character data element known to the stored procedure is parsed, character data element #1 in the example. On a subsequent parse at step **64**, character data elements unknown to the stored procedure are skipped, down to the next occurrence of a character data element known to the stored procedure. In the example, character data element #2 is skipped to the next occurrence of character data element #1.

[0035] The invention has been described with reference to the preferred embodiments. Modifications and alterations will occur to others upon a reading and understanding of the specification. It is our intention to include all such modifications and alterations insofar as they come within the scope of the appended claims, or the equivalents thereof.